



DECLARATION OF RAY SUORSA

I, Ray Suorsa, hereby declare as follows:

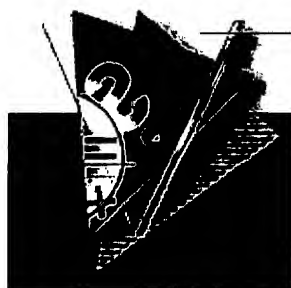
1. I received a M.S. in electrical engineering from Stanford University in 1995 and a B.S. in electrical engineering from the University of California, Santa Barbara, in 1987. Before Opsware Inc., I worked at the Infoseek Corporation designing, implementing and operating many of the components for a competitive Internet Search Engine. Included in my work were custom HTTP proxies, application servers, search caches, and a light-weight distributed database. Previous to Infoseek, I worked at NASA Ames Research Center where I was the lead developer of a real-time computer vision algorithm targeted for semi-autonomous helicopters.
2. I am presently a Vice President and the Chief Architect at Opsware, Inc., a corporation with offices at 599 N. Mathilda Avenue, Sunnyvale, CA 94085.
3. One of ordinary skill in the art of computer science at the time of the invention would have understood that to "expand" a template means to use the template.
4. The attached webpage from astma.it.bond.edu.au/astma=-spec-2.0r1.0.dbk?section=9, although published in 2005, reflects the understanding in the art regarding what one of ordinary skill would understand "expand" a template to mean.
5. The meaning of the term "expand" in the specification of the present application is consistent with the meaning in the art noted above.

I declare under penalty of perjury under the laws of the United States that the foregoing is true and correct. I acknowledge that willful false statements and the like are punishable by fine or imprisonment, or both under 18 U.S.C. §1001 and may jeopardize the validity of any patent granted from the pending application. All statements made herein are of my own knowledge and all statements made herein on information and belief are believed to be true.


DATE:

7/23/07


Ray Suorsa



AsTMa
AUTHORING

<?XML!> <?XML!> 

AsTMa= 2.0 Language Definition

v2.0, 2005-11-24, \$Revision: 1.23 \$

CONTENTS

[Overview](#)

[Authoring](#)

[Updating](#)

[Constraining](#)

[Querying](#)

[Services](#)

[ABOUT](#)

[Faq](#)

[Impressum](#)

[Promotion](#)

[Avatar](#)

9 Templates

During practical map authoring often the situation arises where certain topics all have a structure, be it their type or very similar characteristics. The same is true for stereotypical associations which have to be repeated in a rather monotonous manner.

The language offers a macro facility which processors **MUST** support. These templates are defined first and are then *expanded* to actually render topics, associations, or both. These templates are functions, which implies that they can be parameterized to increase their and reusability.

```
template          →      topic 'return' '[' instance ']'
```

The RETURN clause contains the body of the template.

When a template (function) is declared, it is actually declared as a specialized topic. Templates are NOT part of the generated cTMDM instance; instead they are only kept temporarily for processing and are discarded afterwards. This implies that template identifiers are placed in their own namespace, so that no name clashes with topics in the cTMDM instance can occur.

To parameterize a template, the language understands expressions of the form

```
variable          →      /\$w+/\
```

Such expressions **MUST** be enclosed in {} brackets and can appear in any topic or association declaration as identifiers, part of URIs and part of strings. Any blanks inside the {} pair are ignored.

Any number of such parameters may appear in a template. If none of such appears, the template behaves like a constant.

Example: Template for an association:

```
play-template is-a astma:association-template
return [
  (plays-in-band)
  band      : {$band}
  person    : {$pers} ]
```

Parameters are \$band and \$pers; their names are respectively band and pers.

The language predefines two parameter names, `_left` and `_right`. These can be used of type `astma:infix-association-template` to be placeholders for topic identifiers

9.1 Expanding

Templates must be declared before they can be used. When a template is used, it is said *expanded*. In this process the variables in the template body are replaced with actual values achieved via a binding of parameter names to values or topic identifiers. It is an error if a parameter is not provided with a value.

```
expansion      →      tid 'astma:expands' template-tid [ < binding :
                        { attachment }
```

```
binding →      identifier '=' value
```

Every occurrence of a variable in the template body will then be replaced with the bound value. The resulting topic map fragment will be interpreted as AsTMa= text and the corresponding map content will become part of the constructed cTMDM instance.

Example: Expanding above template into an association:

```
* astma:expands play-template (
band = the-beatles
pers = rho )
```

Example: Expanding above template into an association (using comma separation):

```
* astma:expands play-template (band = the-beatles, pers = rho )
```

For templates of type `astma:infix-association-template` the predefined names `_left` and `_right` are automatically bound when the template is expanded in an *inlined association* (expanding). `_left` will be bound to the topic identifier of the topic which is in the current declaration focus, `_right` will be bound to the topic identifier appearing directly after `_left`.

9.2 Topic Templates

These templates must be of type `astma:infix-topic-template`. Templates of this kind return a single topic. It is an error to provide other topic map content there.

Example: Template for a musician:

```
tmusician is-a astma:topic-template
return [
  {$_left} is-a musician
  bn      : {$name}
  homepage: http://www.wikipedia.org/wiki/{$wp} ]
```

This template is expanded like this:

```
ringo-starr astma:expands tmusician ( name = "Ringo Starr", wp = "Rin
```

cv : http://....

Obviously, more topic attachments can follow the expansion.

9.3 Infix Association Templates

These templates must be of type `astma:infix-association-template`. Templates can only return a single association. It is an error to provide other topic map content that MAY use the predefined names `_left` and `_right` in their body.

AsTMa= processors MUST offer the list of predefined infix association templates (see Appendix C).

Example: An infix association template for a binary relation `is-located-in`:

```
is-located-in is-a astma:infix-association-template
return
  [ (is-located-in)
    location : {$_left}
    region   : {$_right} ]
```

Example: An association template for playing in a band:

```
plays-in-band is-a astma:infix-association-template
return
  [ (plays-in-band)
    musician : {$_left}
    instrument : *
    band      : {$_right} ]
```

Example: An association template for birthdays:

```
born-on is-a astma:infix-association-template
return
  [ (born)
    bio-entity : {$_left}
    birthday   : {$_right} ]
```

9.4 Association Templates

These templates must be of type `astma:association-template`. Templates of this kind return a single association.

9.5 General Templates

These templates must be of type `astma:template`. They can return any topic map content.

Copyright © 2005 Robert Barta Lars Heuer

